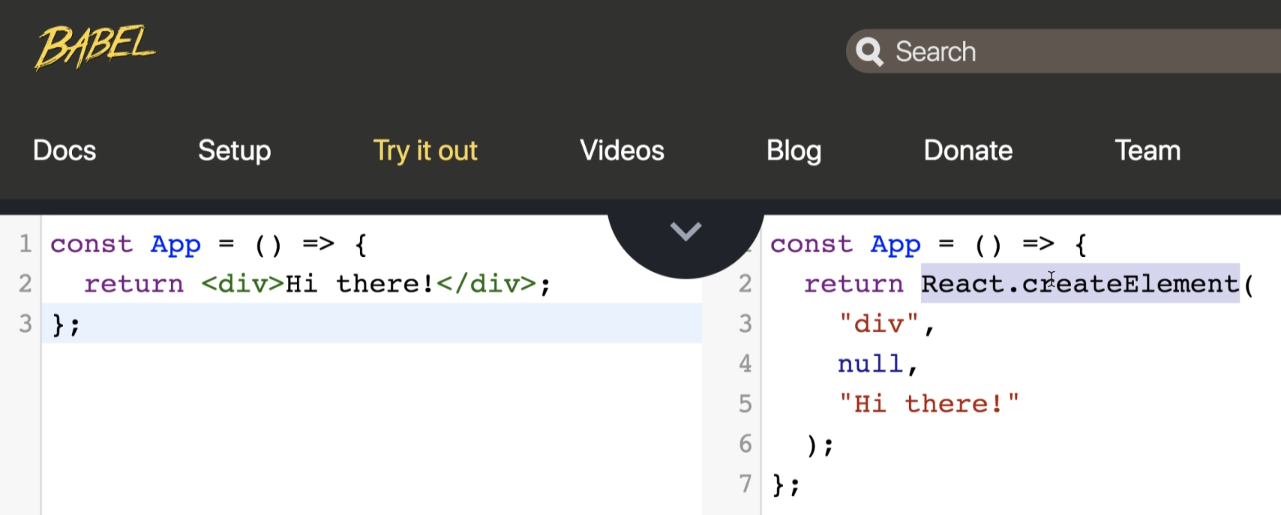
1. **What is JSX exactly?**

Looks like HTML, but is not HTML at all. ES2015 code we write is not directly shipped to the user’s browser (as the browser may not have support for newer versions). In order to get around that, we make use of a dependency named ‘Babel’.

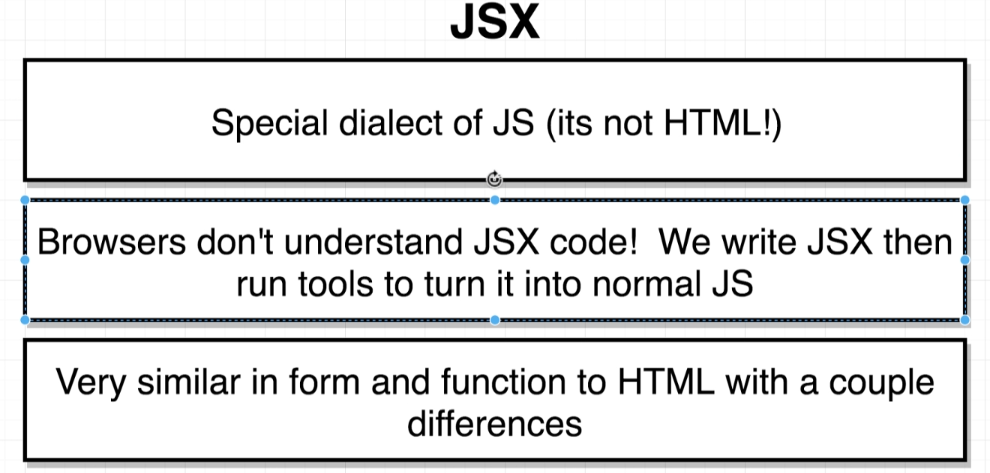
Babel is also used to process the JSX. No browser natively understands what JSX actually is.

Before JSX is sent down to user’s browser, it gets converted into normal looking javascript code.



JSX we write gets converted automatically into function call as above.

Thus, the entire purpose of JSX is to be able to allow us to write out different function calls that normally occur behind the scenes for us in a much easier to understand format (easier to read).



*Note* –

for and id is used so that when we click on the label i.e. the text Enter name : , it automatically selects the input field for the user to write.

<label for = “name” ….

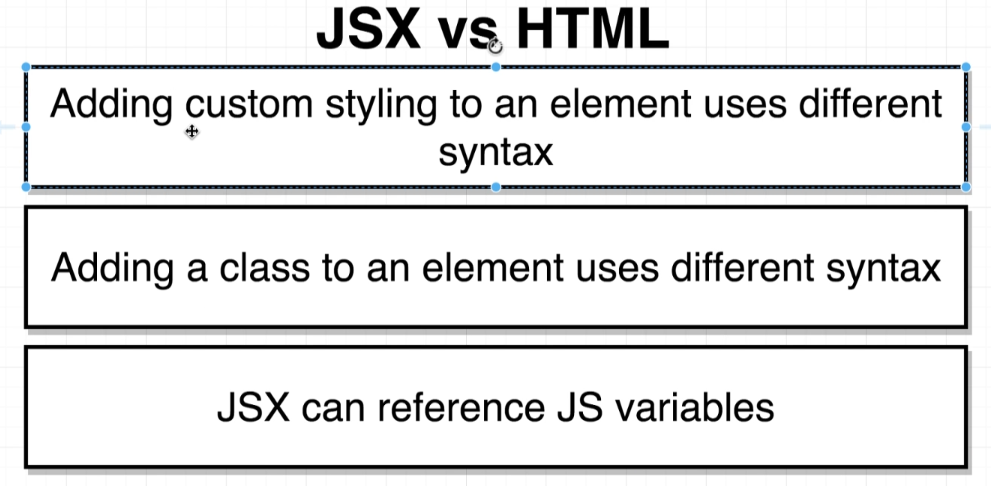
<input id = “name” />

**HTML To JSX –**



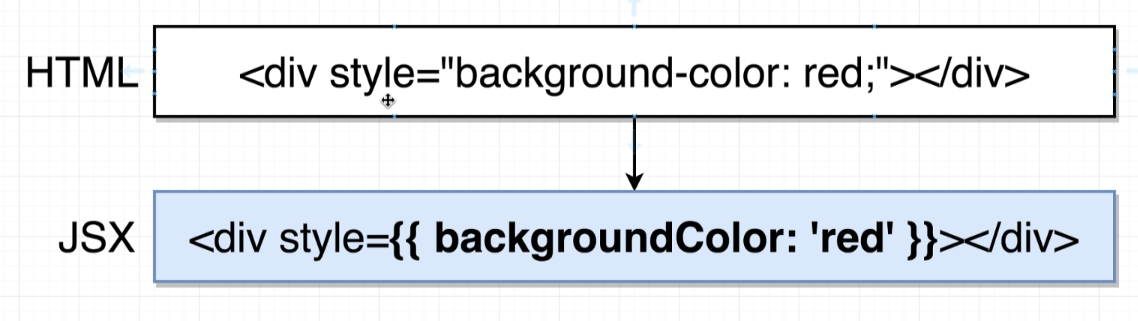
If we paste this exact html into our app component, it gives us an error message.

“The ‘style’ prop expect a mapping from style properties to values, not a string.” (*Error*)



Differences between HTML and JSX

**Inline Styling with JSX –**



**Note 1 –**

Outer curly brace indicates that we want to reference a JS variable inside of our JSX.

The second curly brace is meant to indicate a JS object, wherein the keys reference a different property we want to style and value indicates value for that particular styling. (must be wrapped in a string) background-color : red -> backgroundColor : ‘red’



Convention –

With JSX, we’re supposed to use double quotes anytime we want to indicate a string (like label, name …), but then for any non-JSX property, traditionally we make use of single quotes (blue, red…)

[We can use double quotes also, it’s just a convention to follow]

**Note 2 –**

Anywhere in HTML we assign a class, upon moving to JSX we use a slightly different syntax.

Technically we’re not supposed to use keyword class inside our JSX element. Instead we use c**lassName**.

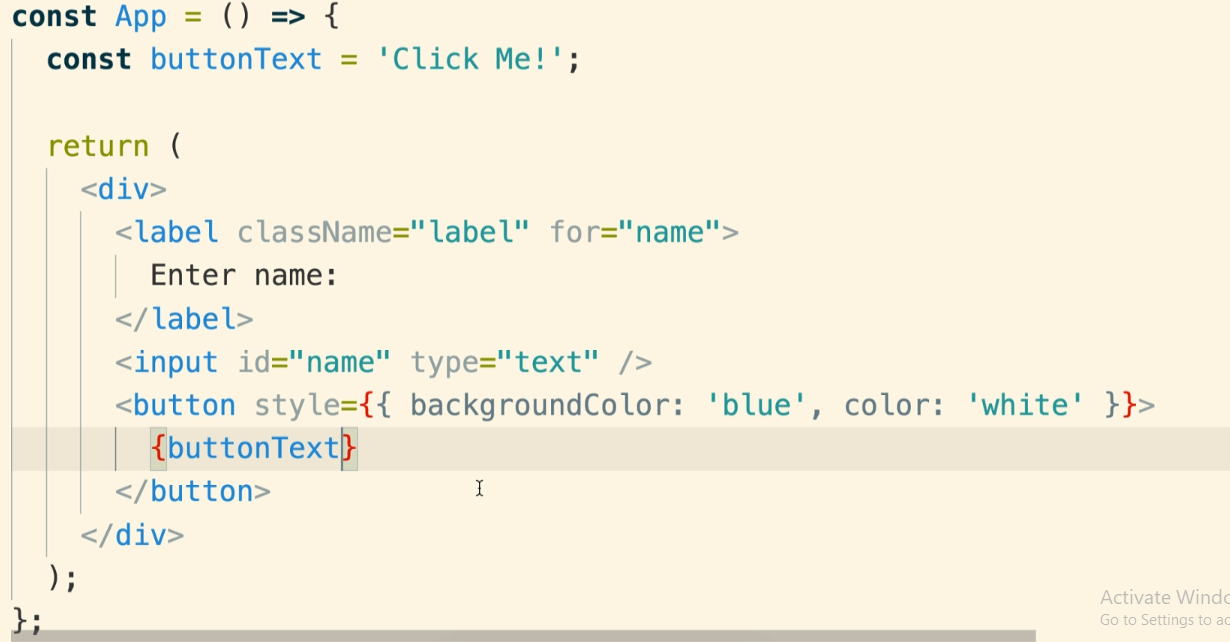
(Just to avoid collisions with keyword of class)



**Note 3 – (most important feature of JSX)**

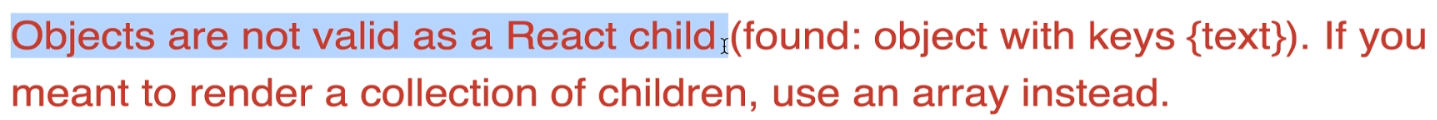
JSX can very easily reference Java Script variables i.e. we can take a JS variable and easily print it inside JSX block.

{} -> does not indicate a JS object, but indicates that we want to reference a JS variable in that place.



Now, there’s a **couple of limitations** around what type of variable we can reference inside JSX.

We can reference string, numbers, and array of string and/or numbers. However, we cannot reference JS objects inside JSX.

Error -

We can reference a specific value of the JS object and reference it. (i.e. {objectName.key})



Technically, we can’t show a JS object as an attribute on an element (say a button)

What we’re really doing here is we’re saying we want to reference a JS variable for this style property and the value that we’re going to pass in is this JS object.

We can also do it like –



**THUS,**

We can use JS objects as long as we’re not trying to print them up as text or something similar.

Upon opening the console in the Browser, we can see an error message up there.



Thus, there are a couple other differences between HTML and JSX. React is going to look at all the different properties you’re passing to elements inside of our JSX block, and if it notices that we’re passing in an invalid property, it’s going to show us a warning message.

Here it means that label property has attribute name of for, but React wants it to be htmlFor.